Data and Node Recovery

MetaArchive Project, Extension Phase

2008-02-08

Summary

This document provides an in-depth guide to LOCKSS data and node recovery activities that may need to be undertaken by Private LOCKSS Networks in case of node failures. The guide provides an overview of the tasks to be undertaken in order to restore a failed node, a failed primary node, and restoring data from a LOCKSS cache. Specifically, it includes information regarding the restoration of hardware, rebuilding of a cache, and recovering a title database and keystore,

Restoring a Lost Node

By design, LOCKSS stores harvested material in a cache directory. LOCKSS will maintain the cache's integrity through peer polling and voting with other nodes on your LOCKSS private network. Occasionally an outside occurrence, such as a power failure, problematic hardware, or a piece of errant coding, can corrupt a file system, leaving the LOCKSS cache in an unusable state. This section is geared towards lost cache recovery techniques.

Hardware/Software fault recovery

- Server Failure: If the operating system (Redhat Enterprise or CentOS) has become inoperable, recovery may require a reinstalling the OS using the *kickstart* configuration file specific to your distributed digital preservation network. The LOCKSS cache itself, if located on an external raid array or a separate data partition, may be unaffected. If the data cache is located on the same disk array as the OS you will be rebuilding the entire system. The following steps may be followed:

    1. Shut down the server and disconnect the storage array
    2. Repair any damaged components or prepare a new server for kickstart
    3. Follow the kickstart instructions from Chapter 6 "LOCKSS installation/Kickstart"
    4. Shut down the server and reconnect the external storage array
    5. Power on both the server and storage array
    6. Run the *make_repository.sh* script to create the data cache volume groups with the command: `/home/lockss/bin/make_repository.sh`
    7. Configure the LOCKSS software by running `/etc/lockss/hostconfig`
    8. Start LOCKSS by issuing the command `/sbin/service lockss start`

- Storage Array Failure: Depending on the severity and nature of the failure, the array may need to be repaired or replaced leaving your cache on empty. After a catastrophic failure, the following steps will assist you in putting your external cache back into play.

    1. Shutdown LOCKSS: `/sbin/service lockss stop`
    2. Un-mount the LOCKSS cache directories with the command: `umount /*cache_directory*` and comment out the cache directory entries in /etc/fstab
    3. Remove the logical volume groups either via the LVM GUI (system-config-lvm) or with the command line tools vgchange, vgremove
        1. `vgchange -a n vg_name`
        2. `vgremove vg_name`

4. Shutdown the array and make any needed repairs
5. Startup and reconfigure the storage system via the array's web utilities
6. Shut down the server, reconnect the storage array, and then power the server on. The new disks should show up in /dev as /dev/sdb through /dev/sdg
7. Use fdisk to format each of the new disk slices: fdisk /dev/sdb; fdisk /dev/sdc, etc.
8. Use the *make_cache.sh* script located in /home/lockss/bin to recreate the volume groups.
9. Start LOCKSS by issuing the command `/sbin/service lockss start`

Archival Units

Archival Units may be likened to a run of journals or a web-based collection containing images, videos, and text. Long-term access to these AU's is insured via LOCKSS harvesting and the polling of loyal peers. The concept of having loyal peers insures that data may be gained from other peer nodes if the original content is no longer available from the publisher.

Archival Unit Configuration

To begin rebuilding your cache, archival units should be added either manually in the LOCKSS administrative interface, or by restoring a cache's archival unit config backup file. Such backups and estores are done at http://address-of-the-node:8081/BatchAuConfig. Restoring from an archival unit configuration file is the quickest and easiest route.

Note: it is good practice for the AU configuration for each node to be backed up on a regular basis, either by each cache's systems administrator, or centrally by the Sysadmin of the primary node, or both.

- Adding Titles (manually)
  1. From the user interface (UI) select "Journal Configuration"
  2. Select "Add Titles"
  3. Choose to add groups of AU's based on node association (individual AU's can also be selected).
- Adding Titles (backup file)
  1. From the UI select "Journal Configuration"
  2. Click on "Restore"
  3. Browse to a previously backed up cache configuration (by default this file is named "BatchAuConfig")
  4. Click on Restore

Speeding up data restoration:

- Manually initiate the crawl process for Aus by going to http://address-of-the-node:8081/DebugPanel. Otherwise LOCKSS determines when to crawl via its internal scheduling algorithm.
- Add "acceleration" settings in to the LOCKSS properties file (lockss.xml) temporarily.
  - org.lockss.poll.defaultPollProbability=100
    - Increases the likelihood of a poll being called to 100% when an AU is being considered
  - org.lockss.poll.pollStarterInterval=10m
    - tells the poll starter to look for AUs to call polls every 10 minutes
  - org.lockss.poll.enablePollStarterThrottle=false
    - disables some throttling of poll starting, only use temporarily!
  - org.lockss.poll.v3.maxSimultaneousV3Pollers=10
    - permits 10 polls to be running simultaneously. CAUTION: if too many are running at once, system resources may run low, and hashes could overrun. May require tweaking to find the correct setting.

To implement a temporary lockss.xml file, edit the original and save to a new, web accessible location. Modify the node's /etc/lockss/config.dat file to point to the new url, and restart the LOCKSS daemon. Revert to the normal settings once data restoration is completed.

Restoring A Lost Primary Node

In order for LOCKSS to harvest and preserve data, certain configuration files must remain available via a primary node. These are the title database and public keystore. As a convenience, these files are stored within the conspectus database and are readily available from any participating LOCKSS node. This section explains the steps required to ensure high availability of the conspectus database and public keystore along with instructions on how to restore them in the event of a disaster.

Preparing for Configuration Node Recovery

The conspectus is no safer in the long term than any other web-based source of information. Hence, the best way to preserve its contents is in the preservation network itself. This metadata should be considered to be as crucial as the actual content of the network and should be preserved as such. The only steps required to preserve it are to create a suitable entry for the conspectus in the conspectus and to harvest it. It should then be preserved alongside all other content.

Recovering the Title Database

In the event that the conspectus database is lost, the title database may be recovered from any node in the preservation network. From the console of any node, the following commands can be used to extract the title database:

- Login to your LOCKSS node either via the console or remotely via SSH
- At the command prompt type:
    1. `export HTTP_PROXY=http://localhost:8080/`
    2. `wget -O lockss.xml http://location-of-conspectus/lockss.xml`
- The commands will extract the lockss.xml configuration file from your LOCKSS cache

Recovering the Keystore

Restoring the public keystore is similar to restoring the title database. From any node in the network, the following commands may be entered:

- Login to your LOCKSS node either via the console or remotely via SSH
- At the command prompt type:
    - `export HTTP_PROXY=http://localhost:8080/`
    - `wget -O lockss.ks http://location-of-conspectus/lockss.ks`
- The commands will extract the lockss.ks public key from your LOCKSS cache

Reconfiguring the Nodes in the Network

Once the title database (conspectus) and public keystore have been recovered from the network, they must be made available to all participating preservation nodes. The first step in making them available is to create a web accessible directory similar to the "plugins" directory created during the kickstart based installation process. The following steps will create this new "config" web folder (commands should be issued via a command prompt/shell):

- First we create the "config" folder in our nodes web root directory
    - `mkdir /var/www/html/config`

Now we will copy our plugin_repository.conf created during the kickstart and make some modifications specific to our new "config" directory

- cd /etc/httpd/conf.d`
- `cp plugin_repository.conf node_config.conf`
- Edit *node_config.conf* and change the Directory location to our “config” directory:
    - <Directory “/var/www/html/**config**”>
- Give apache a SIGHUP (as root) to re-read the configuration files
    - `/usr/sbin/apachectl graceful`

The title database will then need to be reconfigured to point to the proper location of the keystore. Copy the recovered keystore and title database to /var/www/html/config/. Use your favorite editor and open /var/www/html/config/lockss.xml. Find the property named “plugin.keystore” and change the sub-property named “location” to point to the new keystore location. The relevant data is displayed below, with the data that needs to be changed in bold:

<property name=”plugin.keystore”>
<property name=”location” value=http://**new.nodes.address/config/lockss.ks**/>
 <property name=”password” value=”{keystore password}” />
</property>

Once these files are available to the rest of the network, all member nodes will need to be reconfigured with the new title database location. Each node will need to run the “hostconfig” script and only modify the “Configuration URL” setting.

- Run: /etc/lockss/hostconfig
- Modify “Configuration URL” to http://new.nodes.address/config/lockss.ks
- Restart lockss
    1. `/sbin/service lockss stop`
    2. `/sbin/service lockss start`

Restoring data from a LOCKSS cache to its original published location

One of the key benefits of participating in a LOCKSS private network is having reliable, long-term access to your material. When viable access to the original content is no longer available, the LOCKSS cache proxy server can be used to serve its harvested content. This harvested content may be pulled from any node participating in your LOCKSS private network. As we have seen in the previous section, harvested content spans archival units, key configuration files, and plugins.

Viewing cached content

- Login to the LOCKSS user interface
- Select Daemon Status
- Select Archival Units
- A URL listing will contain all files associated with your selected AU. One may view and save individual files and pages. (Most HTML pages will not display properly due to HTML code pointing to relative file locations.)
- To view available plugins and their associated URL, choose “Publisher Plugins” from the pull down menu

Using your LOCKSS node as a cache proxy

A LOCKSS node runs a cache proxy that will serve cached content if the original content is no longer available. If the original content is still available, the LOCKSS node will proxy the request to the original site.

Proxy configuration options

- Proxy Access Control: It is important to restrict access to content contained within your node to either an individual IP address or trusted subnets. The “Proxy Access Control” page available via the UI will allow you to configure a list of allowed and/or denied IP addresses or IP ranges.

- Proxy Options: This page contains proxy options relevant to the audit proxy and Internet Cache Protocol (ICP) server
    1. Audit Proxy: The audit proxy server by default is set to run via port 8082 TCP and will only serve cached content.
    2. ICP Server: The ICP server is used to coordinate a set of cooperating proxy cache servers such as those found in a LOCKSS private network and a popular proxy cache server such as Squid. More information may be found at: http://documents.lockss.org/pub-wiki/IntegratingWithSquid

- Proxy Info: This page is used to obtain proxy configuration information for browsers and other proxies.
    1. EZproxy config fragment: For use with an existing EZproxy server
    2. Generate a dstdomain file for Squid: A Squid proxy server “dstdomain” or destination domain rule
    3. Generate a configuration fragment for Squid: Configuration snippet for use with Squid
    4. PAC File: proxy auto configuration for use in your web browser
    5. Combined PAC file: Allows one to insert your LOCKSS node proxy information into an existing PAC file

Cached AU retrieval and proxy testing

It is important to note the differences between the audit proxy that by default runs over port 8082 and the standard proxy service running over port 8080. The audit proxy will only display content that is cached locally on your node as opposed to the standard proxy service that may proxy you to the original content or another LOCKSS node depending on content availability.

- “wget” command line options to test your proxy.

```
$`export HTTP_PROXY=your.node.edu:8080`
$ `wget -S http://your.collection.edu/AU/`
```

Header response will indicate “Via: * (LOCKSS/jetty)” to indicate success

- Using “wget” to create a mirror of a cached AU.

```
$ `export HTTP_PROXY=your.node.edu:8080`
$ `wget -r http://your.collection.edu/AU/`
```

- Using “wget” to restore a lost plugin

```
$ `export HTTP_PROXY=your.node.edu:8080`
$ `cd /var/www/html/plugins/`
$ `wget http://your.collection.edu/plugins/ExamplePlugin.jar`
```

- Using a proxy automatic configuration (PAC) in your web browser.

1. Allow access to a subnet/individual machines via “Proxy Access Control”
2. Under “Proxy Info” select “PAC File”
3. Copy the generated “PAC File” contents to an available web server
4. Modify your web browser proxy settings to use the PAC file URL

Notes on this document

This document document was prepared by Kyle Fenton, Chris Roddy, and Chris Helms on 2008-02-04 and was completed by the MetaArchive Cooperative in February 2008 as an output of the first network test undertaken during its extension phase contract work with the Library of Congress.